July 2021

CLOUD-NATIVE IPV4 OVER A NATIVE IPV6 CLOUD

Pascal Thubert

Patrick Wetterwald

Jerome Tollet

Ahmed Abdelsalam

Ali Sajassi

CLOUD-NATIVE IPV4 OVER A NATIVE IPV6 CLOUD

AUTHORS:
Pascal Thubert
Patrick Wetterwald
Jerome Tollet
Ahmed Abdelsalam
Ali Sajassi

ABSTRACT

Techniques presented herein provide a variation of Internet Engineering Task Force (IETF) Request For Comments (RFC) 8505 to register stub prefixes to a fabric, and also of RFC 6877 to combine Network Address Translation (NATing) and Internet Protocol (IP) version 4 (v4) to v6 and also v6 to v4 translation. The new operation provides for the ability to transport IPv4 cloud-native application flows between applications and services over an IPv6-only fabric without a change in the containers running the applications and services. The techniques further provide for the ability to load-balance services over the Internet, using a variation of RFC 1794 to obtain a /96 route to the cluster location and RFC 2391 to distribute the load over service instances (e.g., pods).

DETAILED DESCRIPTION

The context of the techniques presented herein involve a disruption to building flat IPv6-only Data Center networks. This model avoids the complexities related to overlays and overlay management. Cloud-native solutions relate to the automated deployment, scaling, and management of containerized applications (apps), for example, using Kubernetes. Support for IPv6 is in its infancy and all deployed applications today rely on IPv4 private addresses for cluster local communication and some encapsulation, such as Virtual eXstensible Local Area Network (VxLAN) for inter-cloud communications.

Providing native IPv6 and IPv6-only communications is a next logical step for cloud networking. Some large enterprises have made that step for their internal networks; however, cloud apps and tooling are lagging way behind. Thus, there exists an immediate opportunity to regain leadership in order to become a cloud networking provider that enables running IPv4 tools, apps, and services as they stand in an IPv6-only cloud.

Consider an example network architecture, as illustrated below in Figure 1.

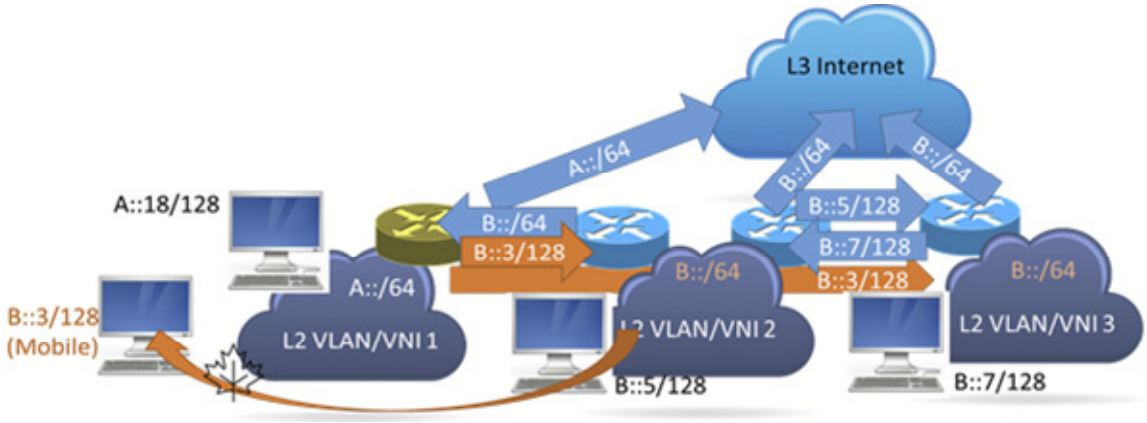

Figure 1: Example Network Architecture

This proposal provides techniques evolved from IPv6 proactive Neighbor Discovery (ND) which is used in Internet-of-Things (IoT) networks and 464XLAT, which is used in IPv6 only Third Generation Partnership Project (3GPP) access, as shown below in Figure 2.

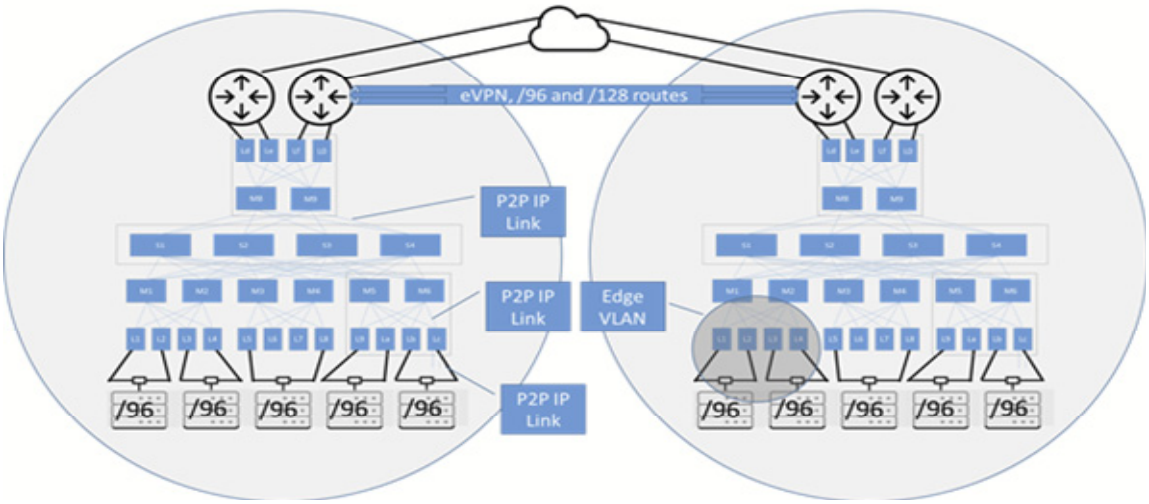

Figure 2: Example Solution Architecture

The techniques of this proposal can be provided via an IPv6 native fat-tree fabric in which any packet is native IPv6, with no encapsulation. Various example steps, discussed below, provide example details for transporting IPv4 packets using a novel variation of the concept of 464XLAT, so that they appear as native IPv6 inside the fabric.

During operation, IP address management (e.g., Kubernetes IP address management (IPAM)) provisions addresses in a virtual router using the Container Network

Interface (CNI) (e.g., Calico), which installs the addresses in the container's interfaces. The IPv4 private address (RFC 1918) can be split in 2 ranges, one for cluster-local (e.g., 192.168.x.y) and one for tenant-global addresses (e.g., 10.a.b.c).

Further, a /96 is provisioned for the cluster. Any IPv4 packet generated inside the cluster going out the cluster is sourced with an IPv6 address. Next, the router is configured as a NAT, cluster-local inside and tenant-global outside. Containers running apps and services typically get cluster-local addresses, so an identical configuration can run in multiple clusters.

Additionally, a variation of RFC 8505 is created for a virtual router / CLAT function (e.g., Calico) used by the virtual router to advertise an IPv6 address (e.g., a /96) to the Top of Rack (ToR) switch an IPv6 prefix and also a mapping with an IPv4 address to its router / ToR switch. Figure 3, below, illustrates example details operations that can be performed in association with the techniques of this proposal.

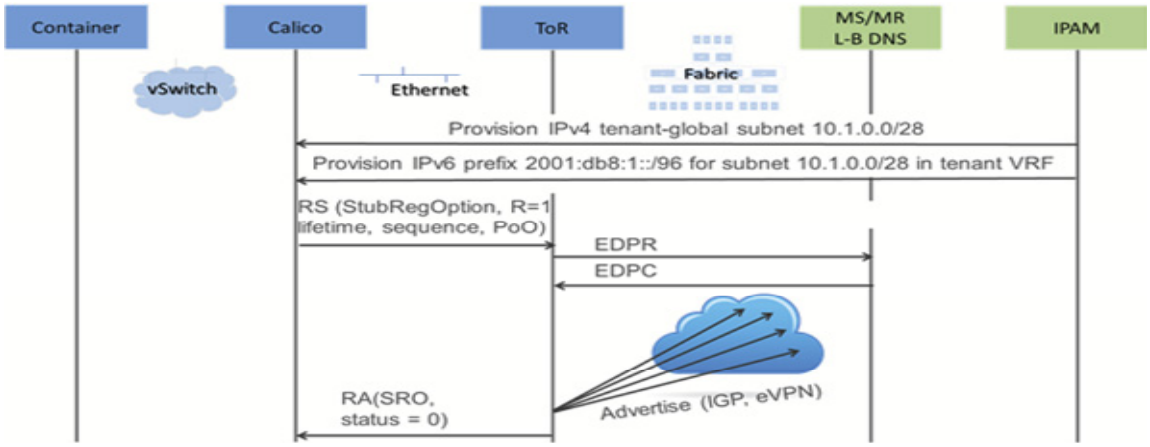

Figure 3: Example Operational Flow

As illustrated above in Figure 2, a new option in the router solicitation (RS) message advertises the stub prefix, e.g., a /96. A dependent sub-option provided for the ability to indicate IPv4 addresses and prefixes that are to be inbedded.

The ToR sets Source Address Validation Improvement (SAVI) (e.g., Access Control List (ACL)) rules to ensure that only the IPv4 addresses declared in the RS message are used as the source in the packets from the virtual router or as destination in the prefix. A new Extended Duplicate Prefix Request/Confirm enables the ToR to register the /96 for

the IPv4 addresses as subnets, when it is desired to reach the IPv4 addresses from the outside, which happens typically for cloud-native services.

Consider various flows, as shown in Figures 4A, 4B, and 4C, which illustrate a packet exchange back and forth between an application on the left and a microservice on the right through which techniques of this proposal may be explained. As shown in Figure 4A, per the art of cloud native operations, the local client looks up a service in the local Domain Name System (DNS), which returns an IPv4 address that indicates the service identifier virtual IP (VIP) address (as opposed to a locator). There is no change from current operations for this step, so cloud-native applications are not modified.

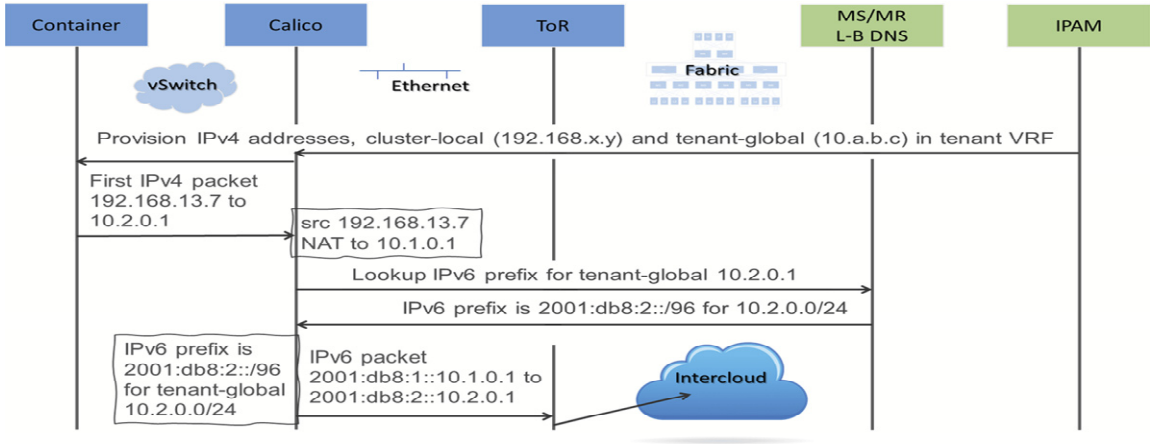

Figure 4A: IPv4 Address Lookup Operations

Next, as shown in Figure 4B, further per the art of cloud native operations, the container-side (C-side) proxy (Calico) intercepts the packet.

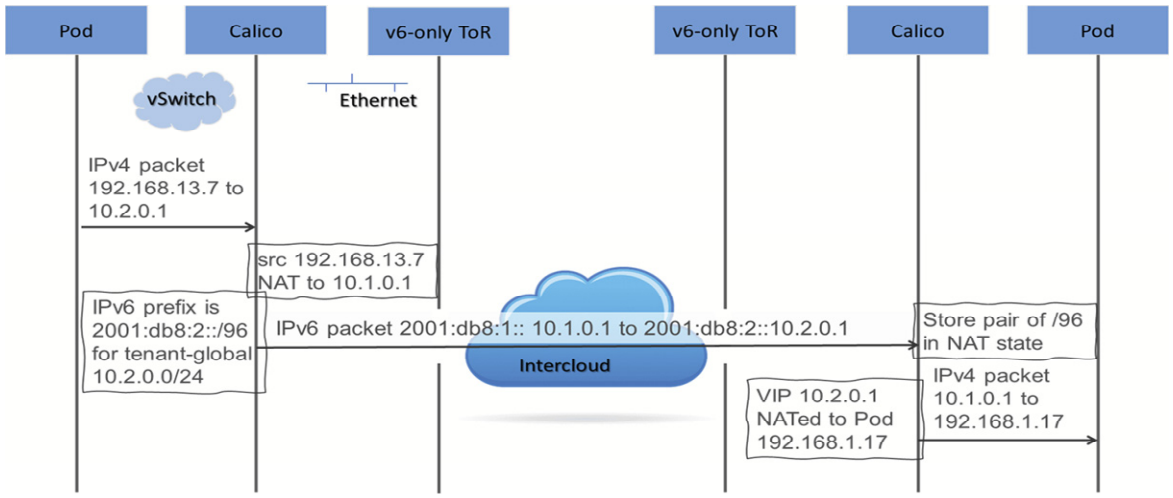

Figure 4B: Packet Interception

Next, as shown in Figure 4C, the (C-side) proxy/Calico performs a new DNS look up to determine the best localization for the service VIP in the destination address. The load-balancing DNS returns the /96 IPv6 prefix of the selected location.

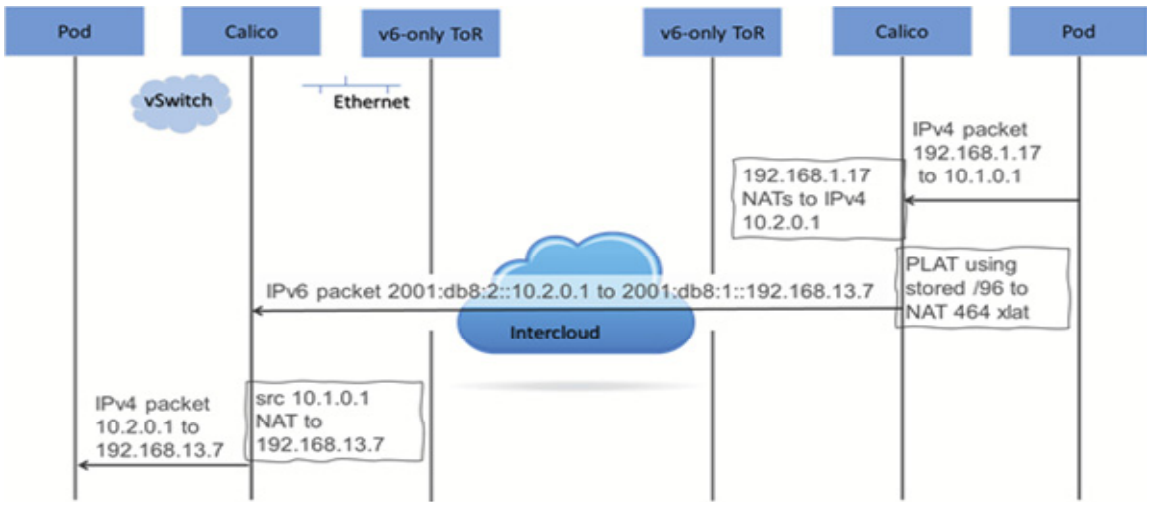

Figure 4C: Facilitating Native IPv6 Packets

The (C-side) proxy/Calico uses that for the novel variation of 464 XLAT, as follows:

- The (C-side) proxy/Calico NATs the IP header IPv4 to IPv6 using the well-known /96 prefix for itself and the /96 prefix discovered above for the destination;
- On the receiving end, the packet is routed to the virtual router that serves the indicated /96, which happens to be a Pod-side (P-side) proxy/Calico;
- As per the art of 464XLAT, the (P-side) proxy/Calico stores the pair of IPv6 /96 prefixes in its NAT state and turns the packet to IPv4;
- As per the art of cloud native: the (P-side) proxy/Calico integrated function selects an IPv4 Pod that runs an instance of the requested service; and
- On the return path, the NATs reverse their forward operation, as in the art of NAT and 464XLAT.

In summary, techniques presented herein provide a variation of RFC 8505 to register stub prefixes to a fabric, and also of RFC 6877 to combine NATing and v4 to v6 and also v6 to v4 translation. The new operation provides for the ability to transport IPv4

cloud-native application flows between applications and services over an IPv6-only fabric via packets are native IPv6 in the fabric without a change in the containers running the applications and services. The techniques further provide for the ability to load-balance services over the Internet, using a variation of RFC 1794 to obtain a /96 route to the cluster location and RFC 2391 to distribute the load over service instances (e.g., pods). Thus, functions such as Transmission Control Protocol (TCP) Segmentation Offload, SAVI, Deep Packet Inspection (DPI), etc. can run anywhere in the fabric.